Optimal Path Planning Based on a Multi-Tree T-RRT* Approach for Robotic Task Planning in Continuous Cost Spaces

Cuebong Wong, Erfu Yang, Xiu-Tian Yan
Space Mechatronic Systems Technology Laboratory
Strathclyde Space Institute
University of Strathclyde, Glasgow G1 1XJ, UK
{cuebong.wong, erfu.yang, x.yan}@strath.ac.uk

Dongbing Gu
School of Computer Science and Electronic Engineering
University of Essex
Wivenhoe Park, Colchester CO4 3SQ, UK
dgu@essex.ac.uk

***Abstract*—This paper presents an integrated approach to robotic task planning in continuous cost spaces. This consists of a low-level path planning phase and a high-level *Planning Domain Definition Language* (PDDL)-based task planning phase. The path planner is based on a multi-tree implementation of the optimal Transition-based Rapidly-exploring Random Tree (T-RRT*) that searches the environment for paths between all pairs of *configuration waypoints*. A method for shortcutting paths based on cost function is also presented. The resulting minimized path costs are then passed to a PDDL planner to solve the high-level task planning problem while optimizing the overall cost of the solution plan. This approach is demonstrated on two scenarios consisting of different cost functions: obstacle clearance in a cluttered environment and elevation in a mountain environment. Preliminary results suggest that significant improvements to path quality can be achieved without significant increase to computation time when compared with a T-RRT-based implementation.**

Keywords - task planning, sampling-based path planning, cost space planning, autonomy, robotics

I. Introduction

There is an ongoing shift towards the deployment of more autonomous robotic systems in complex environments for a diverse range of applications. These span across areas such as plant inspection, planetary exploration, product assembly, building deconstruction, surveillance, search and rescue and much more. All of these tasks require some form of planning to determine how a series of tasks should be performed.

At a high level, the process of planning the sequence in which to perform a series of tasks is called task planning. At a lower level, path planning involves finding an appropriate path between the robot's starting configuration and its final goal configuration. For the most part these two activities have been studied extensively but independently. Recent work have begun to study the unification of path planning and task planning to enable autonomous planning of robots that optimize a solution plan based on the path distances required to reach various landmarks in order to perform some tasks. An example of recent work can be found in [1], where a greedy A*-based path planning algorithm was integrated with a Planning Domain Definition Language (PDDL) planner to tackle the challenges of autonomous robotic exploration missions. However, the use of deterministic graph-search-based path planning methods is unsuited to problems of high dimensions and complexity. Furthermore, little work has been devoted to the process of optimally dealing with general cost spaces in robotic task planning.

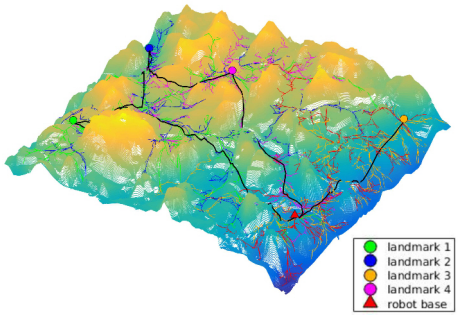


Fig. 1. Example scenario of a task planning problem solved using the multi-tree T-RRT*-based algorithm described in this paper.

To this end, this paper presents an approach to integrated path planning and task planning that deals with continuous cost spaces. The method of path planning is based on a multi-tree extension of the optimal transition-based rapidly-exploring random tree (T-RRT*) [2] to efficiently find an optimal path for all pairs of landmarks. The costs of each path is then passed to a PDDL planner to optimally solve the task planning problem in relation to the path-quality criterion used to compute path costs. This approach is demonstrated on two challenging scenarios. The first involves maximizing obstacle clearance in a cluttered environment, while the second case involves planning on 3-dimensional terrain. A heuristic for path shortcutting of solutions is also presented.

The rest of this paper is organized as follows. A brief review of related work on T-RRT* and PDDL planning is presented in Section II. Then the approach to multi-tree T-RRT* path planning and task planning is described in Section III. Experimental results are presented in Section IV, before a discussion on this approach and future work is provided in Section V. Section VI concludes this paper.

II. Related Work

A. Multi T-RRT* Path Planning

Sampling-based approaches to path planning, such as the probabilistic roadmap [3] and the rapidly-exploring random

tree (RRT) [4], have been widely adopted due to their effectiveness in solving high dimensional problems. Focusing on RRT-based algorithms for its simplicity and efficiency in solving complex planning problems, numerous variants of RRT have been proposed to address the sub-optimality and slow convergence rates associated with the original RRT algorithm. The RRT* [5] provides asymptotic optimality by introducing a rewiring function to iteratively update the path quality to nodes within the tree. The informed RRT* [6] further improves the convergence rate of the RRT* by biasing sampling of new nodes within an elliptical subset of the configuration space determined from the current best solution found.

Meanwhile, bi-directional RRT methods have also emerged. RRT Connect was proposed in [7], which attempted to connect multiple trees grown from different tree roots. This approach demonstrated significant improvements to the convergence rate of the standard RRT for complex problems. A bi-directional RRT* [8] variant was developed to provide the same asymptotic optimality properties of the RRT* method. More recently, the bi-directional tree approach to path planning was successfully demonstrated using the informed RRT* [9], improving both quality of solutions and efficiency of the planner.

All of the above works considered path planning only from the perspective of minimizing path length. They are, however, ineffective for planning in continuous cost spaces. An alternative variant of RRT called Transition-based RRT (T-RRT) [10] was proposed with this problem in mind. The approach introduces a transition test during the sampling stage, which chooses to accept or reject a sample based on the relative differences between the cost of the new sample and the cost of the nearest node within the tree. As a result, samples that lead to low-quality paths would not be added to the tree. A multi-tree extension of the T-RRT variant was later investigated in [11], demonstrating the effectiveness of solving complex continuous cost space problems by growing a tree at each *waypoint* of interest.

Nevertheless, T-RRT do not possess any optimality guarantees, hence solutions are sub-optimal. This was then addressed by combining T-RRT and RRT* [2], bringing together the transition test and near-neighbour wiring procedures of these two variants to guarantee asymptotic optimality while dealing with continuous cost spaces.

B. PDDL Planning

The PDDL was developed in 1998 to standardize the representation of AI planning problems [12], and was used to assess automatic planners in the International Planning Competition (IPC). PDDL captures the definition of a problem and the related physics of a domain through two component files: the *domain* file and the *problem* file. With each IPC the PDDL has been further developed to consider additional features of AI problems, with the latest version being PDDL version 3.1. However, most planners are unable to handle all features that can be expressed through PDDL. To facilitate this, subsets of features are grouped into *requirements*. Hence each domain file must specify the requirements necessary for solving related problems. Consequently, the type of planner used must match the requirements of the planning problem it is applied to.

The PDDL approach to solving planning problems have been applied to a variety of complex problems such as vehicle routing [13] and robot manipulation [14]. In this paper, PDDL is used to formally represent the robot task planning problem with the requirement that the chosen planner used to solve this problem must be capable of handling STRIPS actions [15], actions involving numerical expressions and minimization of plan metrics (the cost representing path quality).

For the requirements described above, an extension of the Local Search for Planning Graphs (LPG) [17] planner, called LPG-*td* [16], was chosen to solve the problems represented in PDDL. It also possesses capabilities to handle much more requirements such as *durative actions*, which includes a duration parameter for each action to handle various consequences at the start, all over or at the end of an action. Nevertheless, there are numerous planners available that also meet the requirements above, such as LAMA (a planner based on pseudo-heuristics derived from landmarks) [18], Metric-Fast Forward (Metric-FF) [19], and Subgoal Partitioning and Resolution in Planning 6 (SGPlan6) [20].

III. General Approach

A. Task Planning Domain

The task planning domain is defined as follows. We assume a single robot free to move in an environment that may consist of obstacles. The robot is required to perform a series of tasks located at n waypoints within the environment. These waypoints represent particular configurations $q_{init}^k, k = 1...n$, within the configuration space C. Tasks may have certain prerequisites or dependency on the completion of other tasks and therefore the sequence in which to perform these tasks may not be arbitrary. The robot starts at some initial waypoint q_{init}^0 and, for each movement from one waypoint to another, the robot accumulates a total cost based on the path cost $c_p(q^i, q^j)$, where q^i represents the waypoint that the robot travels from and q^j is the waypoint the robot is travelling to. This cost is computed from path planning according to predefined path quality criteria. The goal of the task planner is to find a sequence of actions that lead to the completion of all tasks while minimizing the total cost of the plan. This is illustrated in Fig. 2. In order to smoothly relate the information from path planning into the task planning process, a script is developed to automatically generate the required PDDL domain and problem files from within the path planning environment. We use *MATLAB* to execute these processes.

B. Optimal Path Planning in Continuous Cost Space

1) Problem Formulation: Letting C represent the robot configuration space, all infeasible regions due to collisions is denoted as $C_{obs} \subset C$. Consequently, the obstacle-free space is defined as $C_{free} := C \backslash C_{obs}$. For a given initial configuration $q_{init} \in C$ and goal configuration $q_{goal} \in C$, the path planning problem involves finding a feasible path $\sigma : [0, 1] \rightarrow C$ such

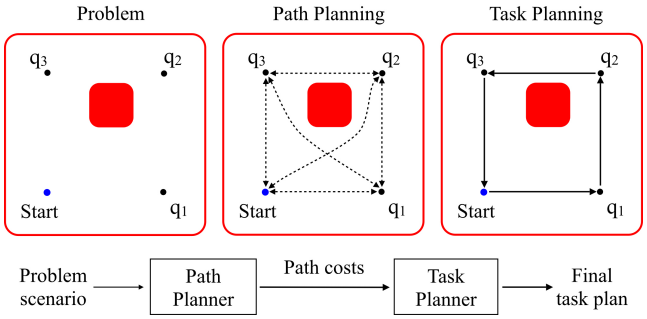


Fig. 2. Illustration of the integrated approach. Given a problem scenario defining task requirements, the path planner computes an optimal path for all pairs of waypoints q_k, including the robot start configuration. The costs of these paths are consolidated in the task planner, which finds an optimal path sequence to meet task requirements.

that $\sigma(\tau) \in C_{free}$ for all $\tau \in [0,1]$. The set of all feasible paths in C is denoted as Σ_{free}.

For continuous cost spaces, the cost function is defined as $c : C \rightarrow \mathbb{R}_+$ such that a real positive cost value is assigned to all configurations within the set C. Hence cost space path planning consists of solving the above path planning problem while taking into consideration the function c during sampling of the configuration space and optimization of the solution path. In order to assess path quality of solutions, a path quality criterion is defined as $c_p : \Sigma_{free} \rightarrow \mathbb{R}_+$ such that a real positive cost value is assigned to every feasible path within C. Using the concept of mechanical work as the quality criterion [2] [10], the cost of any given path is determined by

$$\omega_c \sum_{k=1}^{n} max\left\{0, \left(c\left(q_k\right) - c\left(q_{k-1}\right)\right) d_i\right\} + \omega_d \sum_{k=1}^{n} d_i \quad (1)$$

where d_i is the distance between q_i and q_{i-1}, and w_c and w_d are weights associated with cost space and distance, respectively, and are used to prioritize between finding lower cost paths versus shorter paths. Here w_d is set to a very low value relative to w_c such that the main objective is to find paths with optimal mechanical work values. Hence the goal of planning optimal paths in continuous cost space requires finding a solution path with a minimum path cost c_p.

2) The T-RRT Based Multi-Tree Approach:* The approach developed in this work adopts the heuristics presented in [8] and combines it with the concept of T-RRT* to enable effective connections of multiple trees in an efficient manner. Like all variants of RRT, our approach progressively grows a tree by iteratively sampling the configuration space. The pseudo-code of our approach is presented in Algorithm 1.

First, n trees are initialized and rooted at each waypoint q^k_{init} for $k = 1...n$. During each iteration, a single tree, denoted by T' to identify it from the remaining trees, is chosen for expansion in a round-robin fashion. A random configuration, q_{rand} is then sampled with a small bias towards selecting one of the roots of the other trees as q_{rand}. Like any RRT algorithm, the nearest node $q'_{nearest}$ in T' is identified and a new point q_{new} is generated through a steering function. Using the characteristic *transition test* of the T-RRT method,

Algorithm 1 Multi-Tree T-RRT*

Input: The configuration space C, the cost function $c \rightarrow R_+$ and waypoints $q^k_{init}, k = 1...n$
Output: Trees $T_k, k = 1...n$ and path solutions Σ_{best}

1: **for** $k = 1$ **to** n **do**
2: $T_k \leftarrow initTree(q^k_{init})$
3: **end for**
4: $C_{best} \leftarrow \infty; \Sigma_{best} \leftarrow \emptyset$
5: **while not** $stoppingCriteria(T_k, k = 1...n)$ **do**
6: $T' \leftarrow chooseNextTreeToExpand()$
7: $q_{rand} \leftarrow sampleConfiguration(C)$
8: $q'_{nearest} \leftarrow findNearestNeighbour(T', q_{rand})$
9: $q_{new} \leftarrow steer(q'_{nearest}, q_{rand})$
10: **if** $transitionTest(T', c_{nearest}, c_{new})$ **then**
11: $Q'_{near} \leftarrow findNearNeighbours(T', q_{new})$
12: $L_{near} \leftarrow sortNeighbours(Q'_{near})$
13: $q'_{parent} \leftarrow getParent(L_{near}, q_{new})$
14: $addNodeandEdge(T', q'_{parent}, q_{new})$
15: **for all** $(c_p(q_{new}), q'_{near}, \sigma_{near}) \in L_{near}$ **do**
16: **if** $c_p(q_{new}) + c_p(\sigma_{near}) < c_p(q'_{near})$ **then**
17: $rewire(T', q'_{near}, q_{new})$
18: **end if**
19: **end for**
20: **for all** $T_k \neq T'$ **do**
21: $(\sigma_{sol}, c_p(\sigma_{sol})) \leftarrow connectTrees(T', T_k, q_{new})$
22: **if** $c_p(\sigma_{sol}) \neq null$ **then**
23: $C_{best|T',T_k} \leftarrow c_p(\sigma_{sol}); \Sigma_{best|T',T_k} \leftarrow \sigma_{sol}$
24: **end if**
25: **end for**
26: **end if**
27: **end while**
28: $\Sigma_{best} \leftarrow shortcutting(\Sigma_{best})$

a filtering process is applied to reject samples that lie in high-cost regions. The behaviour of the transition test is characterized by the *temperature* T and a *temperature increase rate* $T_{rate} \in (0,1]$, which controls the level of exploration in high-cost regions. Readers are directed to [10] for details of this function's implementation.

For all accepted q_{new}, the algorithm searches for the set of neighbouring nodes Q'_{near} that lie within a radius r from q_{new}. From among these nodes, the algorithm finds the node that leads to the lowest path cost to q_{new} to serve as its parent. To achieve this in a computationally efficient manner, a sorting technique is adopted from [8] (Line 12 in Algorithm 1). The concept consists of generating a list of cost, configuration and path triplets (c_p, q, σ) for each neighbouring node stored within the parameter L and sorting the elements in L according to path quality. In the original implementation in [8], the cost used was simply the path distance from the tree root to q_{new} through neighbouring node q^i_{near}. Instead, we compute $c_p(q_{new})$ using (1) for each neighbouring node and sort the list L according to this value. This approach minimizes the number of collision checks and connecting procedures required for selecting the parent node of q_{new}. Subsequently,

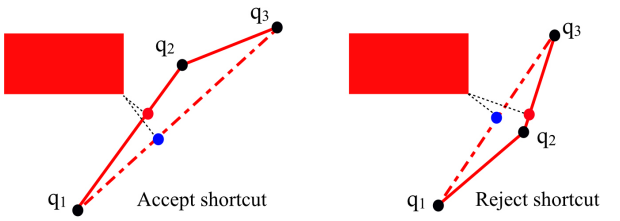


Fig. 3. Illustration of path shortcutting acceptance criteria. Shortcut is accepted only if the maximum cost of a point lying on the original path (shown as a red point) is greater than the maximum cost of a point lying on the shortened path (shown as a blue point). (a) Shortcut is accepted since closest point on new path is further to obstacle than the original path. (b) Shortcut is rejected as the shortened path approaches closer to the obstacle.

the *getParent* function in line 13 performs collision check along the connecting edge to determine if a direct collision-free path exists between q^i_{near} and q_{new}.

Like the RRT*, a rewiring stage is performed to optimize the connections in the tree. For each neighbour in Q'_{near}, a new path cost is computed from the current path cost of q_{new} and the path cost of the connecting edge between said two nodes. Connections to the neighbour node are rewired under the condition that the new path cost is lower than its initial value. Finally, the algorithm checks to see if a connection can be made between T' and every other tree through q_{new}. This is performed using the *connectTrees* function (Line 21), which executes the following: for each tree $T_k \neq T'$, the nearest node $q''_{nearest}$ to q_{new} is found. If the distance between $q''_{nearest}$ and q_{new} is within a predefined step size allowed for tree growth, an attempt is made to connect these two nodes. This check prevents the algorithm from checking for connections if no possibilities exist. Then, all neighbouring nodes of q_{new} in T_k are obtained and a sorting procedure arranges these nodes in order of highest potential for generating a low cost connection. Until a solution is found, the algorithm checks each node to assess if the resulting connection is collision-free and whether improvements to the total path cost $c_p(\sigma_{sol}) < c_{best}$ is observed. When both conditions are true, the connection is accepted and the function returns the solution path and its associated cost function. If no nodes are accepted, the function returns $null$.

Once the termination criteria (line 5) is met, a final *path shortcutting* procedure is executed to optimize the quality of final paths further. Unlike most other variants of RRT where shortcutting simply deals with shortening segments of a solution path that are collision-free, here the procedure must take into consideration the cost function that determines the quality of the path. This is achieved by assessing whether a path shortcut results in an improvement to path quality. This acceptance criteria is implemented by comparing the maximum cost value along the proposed shortcut path against the maximum cost value along the original path. This is illustrated in Fig. 3 for the case of maximizing obstacle clearance. Here a shortcut is accepted only when the closest point along the shortcut σ_{q_1,q_3} to an obstacle is further away than the closest point along the original path σ_{q_1,q_2,q_3} to

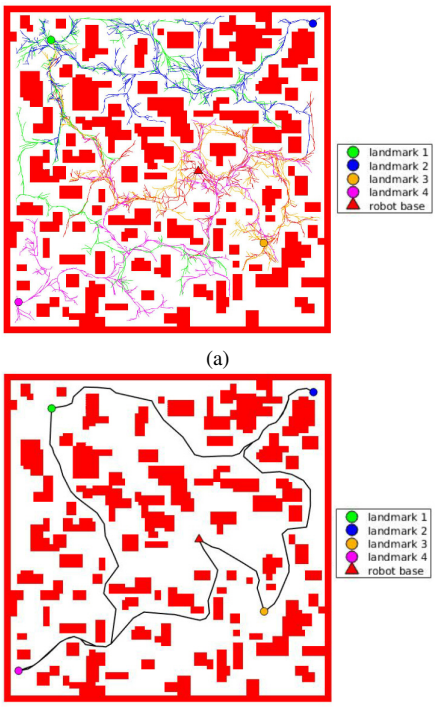


Fig. 4. (a) Growth of five trees in a cluttered environment after 3000 iterations. The colour of each tree corresponds with the color of its tree root shown in the legend on the right. (b) Final path obtained from PDDL planner based on paths found in (a).

an obstacle. Additionally, to account for shorter paths with similar maximal cost, we include a small weighted path length parameter in the comparison as in (1) for computing total path cost. Hence the expression for calculating the cost c_{sc} for shortcutting is given as

$$c_{sc} = \max_{k:q_k \in \sigma_{q_i,q_j}} c(q_k) + \omega_d d_i \tag{2}$$

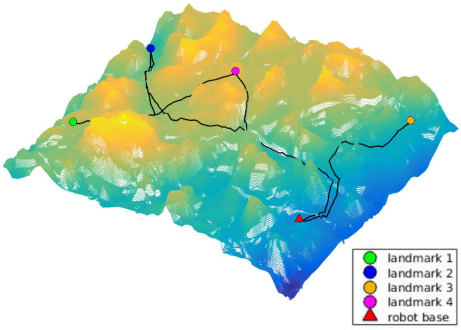
IV. Experimental Results

Our approach was evaluated on two test scenarios consisting of differing types of cost functions. In the first scenario, the integrated task planner was applied to a cluttered two-dimensional environment where the path quality is described by obstacle clearance (Fig. 4). Here the robot starts from base and must visit four landmarks with no constraints on the order in which these landmarks must be visited. The robot must then return to base. Fig. 4a presents the expanded T-RRT* trees rooted at each waypoint for 3000 iterations and Fig. 4b shows the resulting path found from the PDDL planner using path costs obtained from the multi-tree T-RRT* path planner.

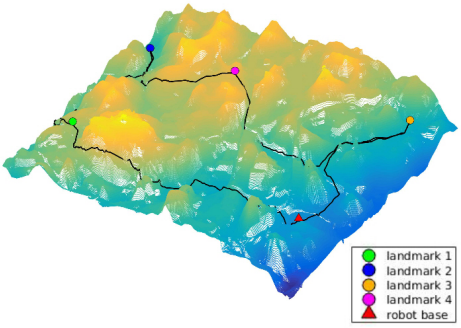
The power of PDDL-based task planning to support diverse types of problems is briefly examined in scenario two, which consists of a 3-dimensional mountain environment. Planning in these environments are typically more challenging for a number of reasons. For one, traditional path planning methods are slow in these environments due to their increased scale. Solutions found by planning algorithms that simply find the shortest path are often far from true desired optimal paths

TABLE I
COMPARISON BETWEEN T-RRT* AND T-RRT-BASED INTEGRATED MULTI-TREE PLANNER

Method	Environment	Total tree nodes	Total path cost	Total path length	Computation time (s)
T-RRT*	Mountain	1918	241.66	746.39	7.15
	Cluttered	1777	15.51	510.73	3.19
T-RRT	Mountain	1985	426.00	793.14	6.28
	Cluttered	1845	20.94	507.79	2.65



(a)

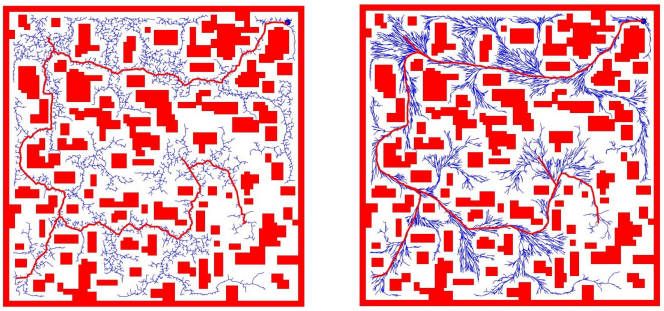


(b)

Fig. 5. Planning in mountain environment. (a) Initial planner solution found after 1188 iterations. Sequence of motion: robot base → landmark 3 → robot base → landmark 4 → landmark 1 → landmark 2 → robot base. (b) Improved planner solution after 3000 iterations. Sequence of motion: robot base → landmark 3 → landmark 4 → landmark 2 → landmark 1 → robot base.

as it can result in the traversal of steep slopes. Ultimately, little work has been invested into task planning within these environments that optimize the true quality of the entire route. To address this, path quality is assessed by elevation in this second scenario, as shown in Fig. 5, with two additional problem requirements defined: (i) landmark 2 should be visited after landmark 3, and (ii) the robot has a limited battery supply and can only be recharged at the robot base. For demonstration purposes, it was assumed that the amount of energy required to move between landmarks is directly proportional to the cost of the path. Fig. 5a presents a sample solution of the initial path found after 1188 iterations, while Fig. 5b presents a lower-cost solution found after 3000 iterations. While the solution shown in Fig. 5a involves revisiting the base to recharge, the improved solution solves the problem without an intermediate stop at robot base. This is a result of the planner finding more optimal paths and consequently reducing energy consumption.

Finally, we compare the performance of our integrated



(a) (b)

Fig. 6. Tree expansions comparison for landmark 2 illustrating path optimality. Paths from landmark 2 to other landmarks shown in red. (a) Solution from T-RRT expansion, (b) solution from T-RRT* expansion.

planner with the performance of an equivalent approach that employs T-RRT for the two scenarios described above. These experimental results were obtained using an Intel Core i5 3320M 2.6 GHz processor with 8 GB RAM. Table I presents the total number of nodes, solution path cost, solution path length and the computation time for each test under the following conditions: 100 trials were performed with the termination criteria for line 5 in Algorithm 1 being met when an initial path is found for all pairs of waypoints. The values given in Table I are hence averages of these trials. An example of the tree expansion for T-RRT and T-RRT* is provided in Fig. 6 for one landmark, with resulting paths to other landmarks found from the searched space of these trees shown in red. A discussion of these results are provided in Section V.

V. DISCUSSION

In Fig. 4a we observe that the combined implementation of the transition test and rewiring procedures characteristic of the T-RRT* method enables the algorithm to explore only safe regions of the environment away from obstacles, while keeping track of the most optimal routes to reach all the sampled configurations in the trees. Through the power of using multiple trees rooted at each waypoint, the algorithm can quickly find low-cost paths between all pairs of waypoints despite the complexity of the cluttered environment. From a more general point of view, our integrated approach effectively links together continuous cost space considerations with high-level task planning processes and performs efficiently in complex environments where traditional graph-search based methods fail to find a solution in reasonable time.

Fig. 5 also demonstrates the potential of our approach for extension to *anytime* applications [21]. The concept of anytime algorithms consists of finding an initial feasible (sub-optimal)

path and continuing to improve upon the solution over time as the robot executes the initial plan. Using our integrated approach, individual solution paths between waypoints can be improved as the robot executes the first path in the task sequence. Furthermore, from a high-level task planning perspective, the order in which waypoints are visited may also be updated based on the improvements made in the path planning phase, leading to changes in how the sequence of tasks are performed online.

In the performance comparison between T-RRT and T-RRT* based approaches presented in Table I, we observe a significant improvement in the total path cost of solutions obtained from T-RRT* for both test scenarios. On average, we observe a 26% improvement for the cluttered environment and a 43% improvement for the mountain environment. These improvements can be explained by the asymptotic optimality guarantee provided by the rewiring process in the T-RRT* method [2]. This has been verified as shown in Fig. 6, where the T-RRT*-based approach provides observably shorter paths than T-RRT for the same set of sampled nodes. Despite these large differences, the T-RRT* based approach does not require significantly longer computation times (20% increase for cluttered environment and 13.8% increase for mountain environment). This result stems from the improved efficiency of the planner due to the implemented heuristics and modifications described in Section III-B. Furthermore, the total number of tree nodes do not differ significantly between the two implementations. This is expected as the filtering behaviour provided by the transition test is unchanged.

While this paper has presented preliminary results for the proposed approach to path and task planning, future work consist of investigating more thoroughly the performance of a unified task and path planner for more complex problems.

VI. Conclusion

In this paper we have presented a new approach to task planning in complex cost spaces through a multi-tree implementation of the T-RRT* algorithm integrated with a PDDL planner. This approach uses a number of heuristics within the rewiring, tree-joining and path shortcutting procedures to generate asymptotically optimal paths without significant increase in computation time when compared with the T-RRT based approach. The planner has been tested on two different scenarios and results demonstrate its potential for extension to an anytime implementation. While experiments discussed in this paper focuses primarily on task planning without constraints, preliminary investigations show that the use of PDDL to model the task planning problem enables expansion to facilitate more complex problems (in a standardized way) involving task-dependencies, numerical constraints and durative actions etc. Ultimately the task problem would only be limited by the capabilities of PDDL and the PDDL solver chosen. Indeed the planner used to solve the PDDL problem need not be LPG-TD as selected in this work, but depends on the nature of the original task planning problem.

Acknowledgment

This research is funded by the Engineering and Physical Sciences Research Council (EPSRC) under its Doctoral Training Partnership Programme (DTP 2016-2017 University of Strathclyde, Glasgow, UK).

References

[1] P. Muoz, M. D. R-Moreno, and D. F. Barrero, Unified framework for path-planning and task-planning for autonomous robots, *Rob. Auton. Syst.*, vol. 82, pp. 114, Aug. 2016.

[2] D. Devaurs, T. Simeon, and J. Cortes, Optimal path planning in complex cost spaces with sampling-based algorithms, *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 415424, Apr. 2016.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566580, 1996.

[4] S. M. LaValle and J. J. Kuffner, Randomized kinodynamic planning, *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378400, May 2001.

[5] S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning, *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846894, Jun. 2011.

[6] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, Informed RRT*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, *in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 29973004.

[7] J. J. Kuffner and S. M. LaValle, RRT-connect: an efficient approach to single-query path planning, *in Proceedings ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, 2000, vol. 2, pp. 9951001.

[8] M. Jordan and A. Perez, Optimal bidirectional rapidly-exploring random trees, *Comput. Sci. Artif. Intell. Lab.*, 2013.

[9] F. Burget, M. Bennewitz, and W. Burgard, BI2RRT*: an efficient sampling-based path planning framework for task-constrained mobile manipulation, *in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 37143721.

[10] L. Jaillet, J. Cortes, and T. Simeon, Transition-based RRT for path planning in continuous cost spaces, *in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 21452150.

[11] D. Devaurs, T. Simeon, and J. Cortes, A multi-tree extension of the transition-based RRT: application to ordering-and-pathfinding problems in continuous cost spaces, *in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 29912996.

[12] D. McDermott, The PDDL planning domain definition language, *AIPS-98 Plan. Compet. Comm.*, 1998.

[13] W. Cheng and Y. Gao, Using PDDL to solve vehicle routing problems, *Int. Conf. Intell. Inf. Process.*, pp. 207215, 2014.

[14] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, Combined task and motion planning through an extensible planner-independent interface layer, *in 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 639646.

[15] R. E. Fikes and N. J. Nhsson, STRIPS: A New Approach to the Application of .Theorem Proving to Problem Solving, *Artif. Intell. 2*, pp. 189208, 1971.

[16] A. Gerevini, A. Saetti, I. Serina, and P. Toninelli, LPG-TD: a fully automated planner for PDDL2.2 domains, *14th Int. Conf. Autom. Plan. Sched. Int. Plan. Compet.*, 2004.

[17] A. Gerevini and I. Serina, LPG: a planner based on local search for planning graphs with action costs, *Am. Assoc. Artif. Intell.*, pp. 1322, 2002.

[18] S. Richter and M. Westphal, The LAMA Planner: guiding cost-based anytime planning with landmarks, *J. Artif. Intell. Res.*, vol. 39, pp. 127-177, 2010.

[19] J. Org Hoomann, The metric-FF planning system: translating ignoring delete lists to numeric state variables, *J. Artiicial Intell. Res.*, vol. 20, pp. 291341, 2003.

[20] Y. Chen, B. W. Wah, and C.-W. Hsu, Temporal planning using subgoal partitioning and resolution in SGPlan, *J. Artif. Intell. Res.*, vol. 26, no. 1, pp. 323369, 2006.

[21] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, Anytime motion planning using the RRT*, *in 2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 14781483.